Puppet show large-scale musical ball and high-intelligent robot large-scale musical ball

●[illegible]---1.[illegible]1[illegible]• [illegible]• [illegible]• [illegible]2[illegible]• [illegible]• [illegible]---2.[illegible]“[illegible]”[illegible]1[illegible]• [illegible]• [illegible]2[illegible]• [illegible]NLP[illegible]• [illegible]3[illegible]• [illegible]• [illegible]4[illegible]• [illegible]• [illegible]---3.[illegible]1[illegible]• [illegible]• [illegible]2[illegible]• [illegible]• [illegible]3[illegible]• [illegible]• [illegible]---4.[illegible]• [illegible]• [illegible]• [illegible]---[illegible]

●[illegible]

[illegible]• [illegible]• [illegible]• [illegible]• [illegible]illusion[illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]“[illegible]”[illegible]• [illegible]• [illegible]

●[illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]

●[illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]

●[illegible]• [illegible] APP [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]LED [illegible]• [illegible]• [illegible]

[illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]

● [illegible]# ## [illegible] 1. **[illegible]** - **[illegible]**[illegible][5](baike.sogou.com/v676260)[10](rili5.com/arc19983/)[illegible] - **[illegible]**[illegible]“[illegible]”[illegible](baike.sogou.com/v676260)(rili5.com/arc19983/)[illegible] - **[illegible]**[illegible][1](jiyifa.com/youer/906392)[9](wenku.baidu.com/view/bc)[illegible]2. **[illegible]** - **[illegible]**[illegible]30 [illegible][5](baike.sogou.com/v676260)[10](rili5.com/arc19983/)[illegible] - **[illegible]**[illegible][6](gxnews.com.cn/staticpag)[11](rili4.com/arc23890/)[illegible] - **[illegible]**[illegible]

[illegible](wenku.baidu.com/view/bc)[illegible]### [illegible] 1. **[illegible]**[illegible](baike.sogou.com/v676260)[12](chinaqw.com/m/hwjy/2025)[illegible]2. **[illegible]**[illegible](gxnews.com.cn/staticpag)(chinaqw.com/m/hwjy/2025)[illegible]### [illegible]“[illegible]”[illegible]1. **[illegible]**[illegible]“[illegible]”[illegible](gxnews.com.cn/staticpag)[12](chinaqw.com/m/hwjy/2025)[illegible]2. **[illegible]**[illegible](chinaqw.com/m/hwjy/2025)[illegible]### [illegible] [illegible]- [illegible]/[illegible]- [illegible]- [illegible]“[illegible]”(jiyifa.com/youer/906392)[9](wenku.baidu.com/view/bc)[illegible]### [illegible] - **[illegible]**[illegible](chinaqw.com/m/hwjy/2025)[illegible]- **[illegible]**[illegible](gxnews.com.cn/staticpag)[illegible]---[illegible](baike.sogou.com/v676260)[illegible](gxnews.com.cn/staticpag)[illegible]

●[illegible]### [illegible] 1. **[illegible]** [illegible] - **[illegible]**[illegible](baike.sogou.com/v676260)[illegible] - **[illegible]**[illegible][5](baike.sogou.com/v676260)[illegible] - **[illegible]**[illegible](baike.sogou.com/v676260)[illegible]2. **[illegible]** - [illegible](jiyifa.com/youer/906392)[2](unjs.com/jiaoan/yinyue/)[illegible] - [illegible](jiyifa.com/youer/906392)[illegible]### [illegible] 1. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]2. **[illegible]** - [illegible][5](baike.sogou.com/v676260)[illegible]3. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### [illegible] 1. **[illegible]** [illegible](unjs.com/jiaoan/yinyue/)[illegible] - [illegible](unjs.com/jiaoan/yinyue/)[illegible] - [illegible]2. **[illegible]** [illegible]### [illegible] 1. **[illegible]** - [illegible][1](jiyifa.com/youer/906392)[illegible] - [illegible](unjs.com/jiaoan/yinyue/)[illegible]2. **[illegible]** - [illegible][2](unjs.com/jiaoan/yinyue/)[illegible] - [illegible][1](jiyifa.com/youer/906392)[illegible]### [illegible] - [illegible][5](baike.sogou.com/v676260)[illegible]- [illegible](jiyifa.com/youer/906392)[illegible][3](subaonet.com/2025/sztw/)[5](baike.sogou.com/v676260)[illegible]

●[illegible]### 1. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]### 2. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 3. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 4. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 5. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 6. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 7. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]

[illegible]### 8. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 9. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 10. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 11. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 12. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### [illegible]

●[illegible]### 1. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 2. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 3. **[illegible]** - **[illegible]**[illegible]### 4. **[illegible]** - **[illegible]**[illegible]### 5. **[illegible]** - **[illegible]**[illegible]### 6. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]### 7. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 8. **[illegible]** - **[illegible]**[illegible]### 9. **[illegible]** - **[illegible]**[illegible]### 10. **[illegible]** - **[illegible]**[illegible]### 11. **[illegible]** - **[illegible]**[illegible]### 12. **[illegible]** - **[illegible]**[illegible]

●[illegible]### 1. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]### 2. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]### 3. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 4. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]### 5. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible]

●[illegible]**[illegible]**[illegible]**[illegible]**[illegible]---### [illegible] #### 1. **[illegible]** - **[illegible]**[illegible]4-6[illegible] - **[illegible]**[illegible]4[illegible](gxnews.com.cn/staticpag)[illegible] - [illegible](gxnews.com.cn/staticpag)[illegible] - **[illegible]**[illegible][2](baike.sogou.com/v676260)[illegible]#### 2. **[illegible]** - **[illegible]**[illegible] - [illegible]1-2[illegible]360°[illegible] - [illegible](baike.sogou.com/v676260)[illegible] - **[illegible]**[illegible] - [illegible]“[illegible]”[illegible] - [illegible][2](baike.sogou.com/v676260)[4](wenku.baidu.com/view/5e)[illegible]####

3. **[illegible]** - [illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]“[illegible]”[illegible][4](wenku.baidu.com/view/5e)[illegible]---### [illegible] #### 1. **[illegible]** - **[illegible]**[illegible] - [illegible] - [illegible][3](gxnews.com.cn/staticpag)[illegible] - **[illegible]**[illegible] - [illegible] - [illegible][2](baike.sogou.com/v676260)[illegible]#### 2. **[illegible]** - **[illegible]**[illegible] - [illegible] - [illegible][4](wenku.baidu.com/view/5e)[illegible] - **[illegible]**[illegible] - [illegible](gxnews.com.cn/staticpag)[illegible]---### [illegible] 1. **[illegible]**[illegible]3D [illegible][4](wenku.baidu.com/view/5e)[illegible]2. **[illegible]**[illegible] - [illegible] - [illegible](gxnews.com.cn/staticpag)[illegible]3. **[illegible]**[illegible](wenku.baidu.com/view/5e)[illegible]---### [illegible] 1. **[illegible]**[illegible]“[illegible]”“[illegible]”[illegible][3](gxnews.com.cn/staticpag)[illegible]2. **[illegible]**[illegible](gxnews.com.cn/staticpag)[illegible]3. **[illegible]**[illegible][2](baike.sogou.com/v676260)[illegible]---### [illegible]“[illegible]”[illegible] 1. **[illegible]**[illegible]2. **[illegible]**[illegible]3. **[illegible]**[illegible]“[illegible]”[illegible](baike.sogou.com/v676260)[4](wenku.baidu.com/view/5e)[illegible]**[illegible]**[illegible](gxnews.com.cn/staticpag)(wenku.baidu.com/view/5e)[illegible]

●[illegible]• [illegible]• [illegible]• [illegible]• [illegible] illusion[illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]“[illegible]”[illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]• [illegible]

●□□□□□□□□□□□□□□□□□□□□□1.□□□□• □□□• □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□WiFi□3G□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□2.□□□□• □□• □□□3.□□□□• □□• □□4.□□□□□• □□“□□”□□□□□□□□□□□□□□□□□□□□• □□□

●□□□□□□□□□**□□□□□□**□**□□□□□□**□□---### **1. □□□□□□□□** #### □□□□□□□□□ ```pythonclass BionicMotionSystem: # □□□ def __init__(self): self.hands = MultiJointManipulator(fingers=5, DOF=20) # □□20□□□□□□ self.legs = DynamicBalancedLegs(sensors=["IMU", "force_feedback"]) self.spine = FlexibleSpine(actuators=12) # □□□□□□□□□□ # □□□□□ def motion_planning(self, task): if task == "□□□□": self.hands.adaptive_grasp(object_shape, force_limit) # □□□□□□□□□□ self.legs.stabilize(zmp_calculation) # □□□□□□□□ elif task == "□□": generate_trajectory(music_beat, style="humanoid") # □□□□□□□□□□ # □□□□□ def realtime_adjust(self): while True: adjust_force = self.hands.tactile_feedback() # □□□□□□□□ self.hands.apply_force(adjust_force) self.legs.correct_posture(self.spine.get_angle()) ```#### **□□□□** - **□□□□□□□**□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ - **□□□□□□**□□□□□□□□ZMP□□□□□□□□□□□□□□□□□ - **□□□□□□**□□□□□□□□□□SMA□□□□□□□□□□□□□□□□□ ---### **2. □□□□□□□□** #### □□□□□□□□□□ ```pythonclass CognitiveNeuralNetwork: # □□□□□□□□□□ def sensory_input(self, vision, audio, touch): self.memory.encode(vision.object_recognition(), audio.speech2text(), touch.pressure_map()) # □□□□□□ def reasoning_engine(self, query): if query.type == "□□□□": return syllogism_solver(query) # □□□□□□□□ elif query.type == "□□□□": return fuzzy_logic(query, context=memory.retrieve()) # □□□□+□□□□□ elif query.type == "□□□□": return neuro_symbolic_integration(query) # □□-□□□□□□□□ # □□□□□□ def emotional_response(self, social_context): emotion_state = empathy_model(social_context.user_emotion) if emotion_state == "□□": self.face.display_tear_effect() # □□□□□□□□□ self.voice.adjust_pitch(-20%) # □□□□ elif emotion_state == "□□": self.motion.dance_style = "celebratory" # □□□□□□□□ ```#### **□□□□□□** | □□□□ | □□□□ | □□□□□□ | |------------------- |---|-----------------------------| | **□□□□** | □□□□□□□□□□□□□“□□□□□□□□□□□□□□□□□□”□ | □□□□□□□□□□□ | | **□□□□** | □□□□□□□□□□□□□□□□□□□□□□ | □□□□□□□□□ | | **□□□□** | □□□□+□□□□□□□□□□□□□□□□ | □□□□□□□□□□□ | | **□□□□** | □□□□□Transformer□GNN□+□□□□□□□□ | □□□□□□□□□□□□□□| | **□□□□** | □□□□□□□if-else□□□□□□□□□ | □□□□□□□□□ | ---### **3. □□□□□□□□□** ```python# □□□□□□□-□□-□□□□ while True: # 1. □□□□ env_data = sensors.capture() # □□LiDAR+□□+□□□□ # 2. □□□□ task_goal = cognitive_network.reasoning_engine(query=env_data.get("user_command"), logic_type="□□□□") # 3. □□□□□ if task_goal.priority == "□□": emotional_response.set("□□", intensity=0.7) motion_system.override_speed(200%) # □□□□□□□□ else: emotional_response.sync(env_data.user_face_expression) # 4. □□□□ motion_system.execute(task_goal.trajectory) ```---### **4. □□□□□□□□□** 1. **□□□□□□**□ - □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ - □□□□□□□□□□□□□□□□□□□□□□ 2. **□

[illegible]-[illegible]**[illegible] - [illegible]“[illegible]”[illegible] - [illegible]“[illegible]”[illegible]“[illegible]”[illegible] 3. **[illegible]**[illegible] - [illegible] - [illegible] ---### **5. [illegible]** ```python def sing_with_emotion(song, emotion): # [illegible] emotion_params = { "[illegible]": {"tempo": +15%, "pitch_variance": 20%, "lyrics_style": "[illegible]"}, "[illegible]": {"tempo": -10%, "vibrato_freq": 5Hz, "lyrics_style": "[illegible]"} } # [illegible] motion_system.lipsync(song.lyrics) # [illegible] motion_system.hands.set_gesture(emotion_params[emotion]["gesture"]) # [illegible] voice.synthesize(song.melody, tempo=emotion_params[emotion]["tempo"], style=emotion_params[emotion]["lyrics_style"]) `````` ---### **[illegible]** [illegible] - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible] [illegible]-[illegible]-[illegible]Neural-Symbolic-Embodied[illegible]AGI[illegible]

● ## [illegible] [illegible](jinchutou.com/shtml/vie)[illegible] ## [illegible] [illegible](houston-tour.com/shengh)[illegible] ## [illegible] [illegible][2](houston-tour.com/shengh)[illegible] ## [illegible] [illegible](njdaily.cn/news/2024/08)[illegible] ## [illegible] [illegible](njdaily.cn/news/2024/08)[illegible] ## [illegible] [illegible](jinchutou.com/shtml/vie)[illegible] ## [illegible][2](houston-tour.com/shengh)[illegible]

● [illegible]

●## [illegible] [illegible]Twitter [illegible](developer.baidu.com/art)[illegible](developer.baidu.com/art)[illegible]## [illegible] [illegible]HTML [illegible](developer.baidu.com/art)[illegible]## [illegible] [illegible]- **[illegible]**[illegible] (developer.baidu.com/art)[illegible]- **[illegible]**[illegible][1] (developer.baidu.com/art)[illegible]- **[illegible]**[illegible] (developer.baidu.com/art)[illegible]## [illegible] [illegible]SVM[illegible]Naive Bayes [illegible]CNN[illegible]RNN [illegible]LSTM [illegible](developer.baidu.com/art)[illegible]## [illegible] [illegible]- **[illegible]**[illegible][1](developer.baidu.com/art)[illegible]- **[illegible]**[illegible][1](developer.baidu.com/art)[illegible]- **[illegible]**[illegible](developer.baidu.com/art)[illegible]## [illegible] [illegible](developer.baidu.com/art)[illegible]

●[illegible]---### 1. **[illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]#### [illegible] ```pythonimport numpy as npdef inverse_kinematics(target_position, initial_joint_angles, link_lengths): # [illegible] # target_position: [illegible] (x, y, z) # initial_joint_angles: [illegible] # link_lengths: [illegible] # [illegible] # [illegible] learning_rate = 0.01 tolerance = 1e-5 max_iterations = 1000 joint_angles = np.array(initial_joint_angles) for i in range(max_iterations): # [illegible] current_position = forward_kinematics(joint_angles, link_lengths) # [illegible] error = target_position - current_position if np.linalg.norm(error) < tolerance: break # [illegible] joint_angles += learning_rate * jacobian_transpose(joint_angles, link_lengths) @ error return joint_angles```---### 2. **[illegible]** - **[illegible]**[illegible]DNN[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]#### [illegible]```pythonimport torchimport torch.nn as nnclass EmotionGenerator(nn.Module): def __init__(self, input_size, hidden_size, output_size): super(EmotionGenerator, self).__init__() self.rnn = nn.GRU(input_size, hidden_size, batch_first=True) self.fc = nn.Linear(hidden_size, output_size) def forward(self, x): # x: [illegible] (batch_size, seq_len, input_size) out, _ = self.rnn(x) out = self.fc(out[:, -1, :]) # [illegible] return torch.sigmoid(out) # [illegible]# [illegible] model = EmotionGenerator(input_size=128, hidden_size=64, output_size=6) # 6 [illegible] input_data = torch.randn(1, 10, 128) # [illegible] 10emotion_probs = model(input_data)print("[illegible]:", emotion_probs)```---### 3. **[illegible]** - **[illegible]**[illegible] Prolog[illegible] - **[illegible]**[illegible] - **[illegible]**[illegible]GNN[illegible] Transformer [illegible]#### [illegible]```pythonfrom sympy import symbols, Implies, satisfiable# [illegible] P, Q = symbols('P Q')# [illegible] rule = Implies(P, Q)# [illegible] print("[illegible]:", satisfiable(rule))```---### 4. **[illegible]** - **[illegible]NLP[illegible]**[illegible] Transformer [illegible] GPT[illegible] - **[illegible]**[illegible] TTS[illegible]Text-to-Speech[illegible]#### [illegible]TTS [illegible]```pythonfrom gtts import gTTSimport ostext = "[illegible]"tts = gTTS(text, lang='zh-cn')tts.save("output.mp3")os.system("start output.mp3") # [illegible]```---### 5. **[illegible]** - **[illegible]**[illegible]Behavior Tree[illegible] - **[illegible]**[illegible]#### [illegible]```pythonfrom py_trees import Behaviour, Blackboard, Statusclass Action(Behaviour): def __init__(self, name): super(Action, self).__init__(name) def update(self): print(f"[illegible]: {self.name}") return Status.SUCCESS# [illegible] root = Action("[illegible]")child1 = Action("[illegible] 1")child2 = Action("[illegible] 2")root.add_child(child1)root.add_child(child2)# [illegible] root.tick_once()```---### 6. **[illegible]** - **[illegible]GNN[illegible]**[illegible] - **Transformer**[illegible]#### [illegible]Transformer [illegible]```pythonimport torchimport torch.nn as nnclass LogicTransformer(nn.Module): def __init__(self, input_size, hidden_size, num_layers, num_heads): super(LogicTransformer, self).__init__() self.encoder = nn.TransformerEncoder(nn.TransformerEncoderLayer(d_model=input_size, nhead=num_heads), num_layers=num_layers) self.fc = nn.Linear(input_size, 1) def forward(self, x): x = self.encoder(x) x = self.fc(x[:, -1, :]) # [illegible] return torch.sigmoid(x)# [illegible] model = LogicTransformer(input_size=64, hidden_size=128, num_layers=4, num_heads=8)input_data = torch.randn(1, 10, 64) # [illegible] 10output = model(input_data)print("[illegible]:", output)```---[illegible]

●[illegible]---### **1. [illegible]** - **[illegible]**[illegible] - [illegible] - [illegible] - **[illegible]**[illegible] - [illegible]Inverse Kinematics, IK[illegible] - [illegible]Reinforcement Learning[illegible] - **[illegible]**[illegible] - [illegible]IMU[illegible] - [illegible]---### **2. [illegible]** - **[illegible]**[illegible] - [illegible] - [illegible] CNN[illegible]Transformer[illegible] - **[illegible]**[illegible] - [illegible] - [illegible]Symbolic AI[illegible] - **[illegible]**[illegible]

 - [illegible]Affective Computing[illegible] - [illegible]TTS[illegible]--- ### **3. [illegible]** - **[illegible]**[illegible] - [illegible]“[illegible] A [illegible] B”[illegible] - **[illegible]**[illegible] - [illegible] - **[illegible]**[illegible] - [illegible] - **[illegible]**[illegible] - [illegible] - **[illegible]**[illegible] - [illegible]--- ### **4. [illegible]** - **[illegible]NLP[illegible]**[illegible] - [illegible] GPT[illegible]BERT[illegible] - [illegible] - **[illegible]**[illegible] - [illegible] WaveNet[illegible]Diffusion Models[illegible] - [illegible]--- ### **5. [illegible]** - **[illegible]**[illegible] - [illegible]Emotion State Machine[illegible] - **[illegible]**[illegible] - [illegible]Imitation Learning[illegible] - **[illegible]**[illegible] - [illegible]--- ### **6. [illegible]**```pythonclass HighIntelligenceRobot: def __init__(self): self.brain = NeuralNetwork() self.motion_controller = MotionController() self.emotion_engine = EmotionEngine() self.language_processor = LanguageProcessor() def perceive(self, sensor_data): # [illegible] visual_data = sensor_data['camera'] audio_data = sensor_data['microphone'] return self.brain.process(visual_data, audio_data) def reason(self, perception): # [illegible] if self.brain.formal_logic(perception): return self.brain.deepen_logic(perception) return self.brain.preliminary_logic(perception) def act(self, decision): # [illegible] self.motion_controller.move(decision) self.emotion_engine.express(decision) def interact(self, human_input): # [illegible] response = self.language_processor.generate_response(human_input) self.act(response)# [illegible] robot = HighIntelligenceRobot()sensor_data = {'camera': 'image_data', 'microphone': 'audio_data'}robot.perceive(sensor_data)decision = robot.reason(perception)robot.act(decision)robot.interact("Hello, how are you?")```--- ### **7. [illegible]** - **[illegible]**[illegible] - **[illegible]**[illegible] - **[illegible] AI**[illegible] - **[illegible]**[illegible]

●[illegible]1.[illegible]1[illegible]2[illegible]VLM[illegible]3[illegible]-[illegible]-[illegible]4[illegible]Markov [illegible]IBM Watson [illegible]2.[illegible]1[illegible]DeepMind [illegible] AlphaGo [illegible]2[illegible]3[illegible]4[illegible]5[illegible]3.[illegible]1[illegible]2[illegible]“[illegible]”[illegible]3[illegible]

●[illegible]---1.[illegible]• [illegible] OpenCV[illegible]Vision Transformer[illegible]• [illegible]LLM[illegible] ChatGPT[illegible]---2.[illegible]1[illegible]ViLaIn [illegible] ViLaIn [illegible]Problem Description,PD[illegible]• [illegible]• [illegible]• [illegible]2[illegible]RoboFlamingo [illegible] RoboFlamingo [illegible] OpenFlamingo[illegible]3[illegible]RT-2 [illegible] RT-2 [illegible]-[illegible]-[illegible]---3.[illegible]1[illegible]Mercury

X1 [illegible] Mercury X1 [illegible] SLAM[illegible]ROS[illegible]OpenCV [illegible] LLM [illegible]• [illegible] SLAM [illegible]• [illegible] OpenCV [illegible]• [illegible] LLM [illegible] [illegible]2[illegible]RoboFlamingo [illegible] RoboFlamingo [illegible]---4.[illegible]• [illegible]• [illegible]• [illegible]---[illegible]

●

[illegible]**[illegible]**[illegible]**[illegible]**[illegible]Neural-Symbolic AI[illegible]

---

1. [illegible]
```python
# [illegible]
class HyperIntelligentRobot:
def __init__(self):
# [illegible]
self.body = BionicBodySystem() # [illegible]
self.sensors = MultiModalSensors() # [illegible]

# [illegible]
self.brain = NeuroSymbolicBrain() # [illegible]-[illegible]
self.emotion = AffectiveEngine() # [illegible]

# [illegible]
self.actuator = DynamicActuator() # [illegible]
self.communication = SocialInterface() # [illegible]

# [illegible]
def run(self):
while True:
# [illegible]-[illegible]-[illegible]
perception = self.sensors.capture()
cognition = self.brain.process(perception)
action = self.actuator.execute(cognition)
self.communication.feedback(action)
```

---

2. [illegible]
[illegible] 1[illegible]
```python
class BionicHand:
def __init__(self):
self.fingers = [
FingerJoint(dof=4, material="shape_memory_alloy"), # [illegible] 4 [illegible]
...
]
self.tactile_sensors = TactileArray(resolution="0.1mm") # [illegible]
```

~~~~
# □□□□□□□
def adaptive_grasp(self, object_properties):
# □□□□□□□□□□□□□□
force_model = ReinforcementLearning(
state=object_properties.shape + self.tactile_sensors.read(),
action_space=self._calculate_grasp_poses()
)
optimal_pose = force_model.optimize()
self._apply_pose(optimal_pose)
```

#### □□ 2□□□□□□□
```python
class BionicLegs:
def dynamic_walk(self, terrain_map):
# □□□□(ZMP)□□□□□□□□□
zmp_trajectory = ZMPPlanner(terrain_map).generate()
nn_correction = NeuralNetworkBalancer(
input=IMU_data + zmp_trajectory,
output="joint_torques"
).predict()
self.joints.apply_torque(nn_correction)
```

---

### **3. □□□□□□□□□**
#### □□ 1□□□-□□□□□□□□
```python
class NeuroSymbolicBrain:
def process(self, perception):
# □□□□□□□
fused_data = MultimodalFusion(
vision=perception.camera,
audio=perception.microphone,
touch=perception.tactile
).encode()

# □□□□□□
logic_output = {
"□□□□": FormalLogicSolver(fused_data).deduce(), # □□□□□□□□
"□□□□": ProbabilisticGraphModel(fused_data).infer(), # □□□□□
"□□□□": NeuralSymbolicIntegrator(fused_data).run() # Transformer+GNN
}
return self._consensus(logic_output) # □□□□□□□□□
```

#### □□ 2□□□□□□
```python
class AffectiveEngine:
def __init__(self):
self.emotion_map = {
"□□": EmotionState(arousal=0.8, valence=0.9),
~~~~

```python
"□□": EmotionState(arousal=0.3, valence=-0.7),
...
}

def update(self, social_context):
# □□□□□□□□□
current_emotion = MultimodalAffectModel(
speech_tone=social_context.voice_analysis(),
facial_expression=social_context.face_recognition(),
semantic_analysis=NLU(social_context.dialog)
).predict()
self._apply_physiological_response(current_emotion) # □□□□/□□□□
```

---

4. □□□□□□□□□
□□ 1□□□□□□□□□
```python
class LanguageProcessor:
def analyze(self, text):
# □□□□□□□□
pipeline = [
("□□□□", SymbolicParser()), # □□□□□□
("□□□□", NeuralEntityExtractor()), # BERT+□□□□
("□□□□", AffectiveClassifier()), # □□□□
("□□□□", ConsistencyChecker()) # □□□□
]
return Pipeline(pipeline).process(text)
```

□□ 2□□□□□□□□□
□□□□	□□□□	□□□□
□□□□	□□□□□□ + □□□□	□□□□□□
□□□□	□□□□□ + □□□□□□□□	□□□□/□□□□
□□□□	□□□□ + □□□□□□	□□□□
□□□□	Transformer + □□□□□(GNN)	□□□□□□□
□□□□	□□□□ + □□□□□□(CSP)	□□□□□□

---

5. □□□□□□□□
```mermaid
graph TD
A[□□□□□□] --> B[□□□□□]
B --> C{□□□□□□}
C -->|□□□□| D[□□□□□□]
C -->|□□□□| E[□□□□□□□□]
C -->|□□□□| F[□□□□□]
D & E & F --> G[□□□□□]
G --> H[□□□□□]
H --> I[□□□□/□□□□]
I --> J[□□□□]
```

J --> A

---

6. □□□□□□□□□□□
1. **□□□□□□□**
- **□□**: □□-□□-□□□□□□□□□
- **□□**: □□□□□□□□(SNN)□□□□□□□□□□□

2. **□□□□□□**
- **□□**: □□□□□□□□□□□□□
- **□□**: □□□□(Hypergraph)□□□□□□□□□□□

3. **□□□□□□□**
- **□□**: □□□□□□□□□□□□
- **□□**: □□"□□-□□□□□□□"□
```python
emotion_motion_map = {
"□□": {"gait_stiffness": 0.9, "hand_gesture": "□□", "eye_led_color": "□□"},
"□□": {"head_movement": "□□□□", "pupil_dilation": 1.2}
}
```

---

7. □□□□□□□
1. **□□□**□□□□□□□□ + □□□□□□□□
2. **□□□**□□□□□□□ + □□□□□□□□
3. **□□□**□□□□□□□□□□□□□□□□□□□□□□□□□□
4. **□□□□**□
```plaintext
□□□□□□□□ □□□□□□□□ □□□□□□□ □□□□□□(AGI)
```

□□□□□□□□□□□□□□□□□Boston Dynamics Atlas+ NVIDIA Jetson AGX□□□□□□□□□□□□□□**□□□□□□□□□□□□□□**□**□□□□□-□□□□**□□□□

□□□□□□□□□□

□□□□□Emotional Computing□□1997□□□□□□□□□□□□MIT□□□□□□□Picard□□□□[1](blog.csdn.net/cf2SudS8x)□□□

□□□□□□□□□□□

□□□□

□□

- **□□□□□□**□□□□□□□□□□□□□□□□□□□□□□□□

- **[illegible]**[illegible]
- **[illegible]**[illegible][1](https://blog.csdn.net/cf2SudS8x8F0v/article/details/80211461)[illegible]

[illegible]

[illegible][1](https://blog.csdn.net/cf2SudS8x8F0v/article/details/80211461)[illegible]

[illegible]

[illegible][1](https://blog.csdn.net/cf2SudS8x8F0v/article/details/80211461)[illegible]

[illegible]

[illegible][1](https://blog.csdn.net/cf2SudS8x8F0v/article/details/80211461)[illegible]

[illegible]

[illegible][1](https://blog.csdn.net/cf2SudS8x8F0v/article/details/80211461)[illegible]

[illegible]

[illegible][1](https://blog.csdn.net/cf2SudS8x8F0v/article/details/80211461)[illegible]

[illegible]

●[illegible]

1.[illegible]
[illegible]

1. [illegible]

2. [illegible]

3. [illegible]

4. [illegible]

5. [illegible]

6. [illegible]

2.[illegible]

（1[illegible]

[illegible]

```python
import cv2 # OpenCV for vision
import speech_recognition as sr # Speech recognition
import numpy as np

class PerceptionModule:
def __init__(self):
self.camera = cv2.VideoCapture(0) # Initialize camera
self.recognizer = sr.Recognizer() # Initialize speech recognizer

def get_visual_input(self):
ret, frame = self.camera.read()
if ret:
return frame
return None

def get_audio_input(self):
with sr.Microphone() as source:
audio = self.recognizer.listen(source)
try:
text = self.recognizer.recognize_google(audio)
return text
except sr.UnknownValueError:
return "Unknown audio"
```

（2[illegible]

[illegible]

```python
import numpy as np

class MotionControlModule:
def __init__(self):
self.joints = np.zeros(20) # Example: 20 joints

def move_joint(self, joint_id, angle):
self.joints[joint_id] = angle
```

```
print(f"Joint {joint_id} moved to {angle} degrees")

def perform_action(self, action):
if action == "wave":
self.move_joint(5, 90) # Example: Move joint 5 to 90 degrees
```

[illegible]3[illegible]

[illegible] TensorFlow [illegible] PyTorch[illegible]

```python
import tensorflow as tf

class NeuralNetworkModule:
def __init__(self):
self.model = tf.keras.models.load_model("path_to_model.h5")

def predict_emotion(self, input_data):
return self.model.predict(input_data)
```

[illegible]4[illegible]

[illegible] NLTK [illegible] spaCy[illegible] GPT-3[illegible]

```python
import openai

class LanguageProcessingModule:
def __init__(self, api_key):
openai.api_key = api_key

def generate_response(self, prompt):
response = openai.Completion.create(
engine="text-davinci-003",
prompt=prompt,
max_tokens=50
)
return response.choices[0].text.strip()
```

[illegible]5[illegible]

[illegible]

```python
class EmotionBehaviorModule:
```

```
def __init__(self):
self.emotions = {"happy": 0, "sad": 0, "angry": 0}

def update_emotion(self, emotion, value):
self.emotions[emotion] = value

def choose_behavior(self):
if self.emotions["happy"] > 0.5:
return "wave"
elif self.emotions["angry"] > 0.5:
return "cross_arms"
return "stand_still"
```

□6□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```python
class DecisionPlanningModule:
def __init__(self):
self.tasks = []

def add_task(self, task):
self.tasks.append(task)

def execute_tasks(self):
for task in self.tasks:
print(f"Executing task: {task}")
# Example: Call other modules to perform the task
```

3.□□□□□
□□□□□□□□□□□□□□□□□

```python
class RobotSystem:
def __init__(self):
self.perception = PerceptionModule()
self.motion = MotionControlModule()
self.neural_net = NeuralNetworkModule()
self.language = LanguageProcessingModule("your_api_key")
self.emotion_behavior = EmotionBehaviorModule()
self.decision_planning = DecisionPlanningModule()
```

~~~
def run(self):
while True:
visual_input = self.perception.get_visual_input()
audio_input = self.perception.get_audio_input()

# Process inputs
emotion = self.neural_net.predict_emotion(visual_input)
response = self.language.generate_response(audio_input)

# Update emotion and choose behavior
self.emotion_behavior.update_emotion("happy", emotion[0])
behavior = self.emotion_behavior.choose_behavior()

# Execute behavior
self.motion.perform_action(behavior)

# Plan and execute tasks
self.decision_planning.add_task(response)
self.decision_planning.execute_tasks()
```
~~~

4.[illegible]

[illegible]

●[illegible]

1.[illegible]

[illegible]

[illegible]1[illegible]

[illegible] RNN[illegible]LSTM[illegible]Transformer [illegible] BERT[illegible]BERT [illegible]

[illegible]2[illegible]

[illegible] CNN [illegible] RNN[illegible]

[illegible]3[illegible]

[illegible]CNN[illegible]

2.[illegible]

[illegible]

□1□□□□□
□□□□□□□□□□Transformer□LSTM□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“□□”□□□□
□□□□□□□□□□□□□□□□

□2□□□□□
□□□□□□□□□□□□GPT-3□□□□□□□□□□□□□□□□□□□□□“□□”□□□□□□□□□□□□□□□□□

3.□□□□□□□
□□□

□1□□□□□
□□“□□□□”□

□2□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

4.□□□□□□
□□□□□□□BERT□□□□□□□□□□□□□□□□□□

```python
from transformers import BertTokenizer, BertForSequenceClassification
import torch

# □□□□□□BERT□□□□□□
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',
num_labels=5) # □□□5□□□□□

# □□□□
text = "I am very happy today!"
inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True,
max_length=512)

# □□□□
with torch.no_grad():
outputs = model(**inputs)
predicted_emotion = torch.argmax(outputs.logits, dim=1).item()

# □□□□□□□□
emotion_to_behavior = {
0: "angry", # □□0□□□□
1: "sad", # □□1□□□□
2: "happy", # □□2□□□□
3: "surprised", # □□3□□□□
4: "neutral" # □□4□□□□
}
```

```
behavior = emotion_to_behavior[predicted_emotion]
print(f"Detected emotion: {behavior}")
```

5.□□
□□□□□□□□□□ BERT□LSTM□Transformer□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□

●□□**□□□□□□□□□□□□□□□□□**□□**□
□□□□□□**□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Python □□□□□□□□□□□□□

---

1. □□□□Physical Layer□
□□□□□□□□□
```python
class MotorController:
def __init__(self, joint_limits):
self.joint_angles = {joint: 0 for joint in joint_limits} # □□□□□□□

def inverse_kinematics(self, target_pos):
# □□□□□□□□□□□□□□□□□□□
# □□□□□□□□□□□□□□□□□□□□
pass

def move_joint(self, joint, angle):
# □□□□□□□□□PID □□□□□□□□□□□
if angle within joint_limits[joint]:
self.joint_angles[joint] = angle
send_command_to_motor(joint, angle)
```

---

2. □□□□Sensory Layer□
□□□□□□□□
```python
class SensorFusion:
def __init__(self):
self.camera = VisionSystem()
self.microphone = AudioProcessor()
self.touch = TactileSensor()
self.olfactory = SmellSensor()

def update(self):
# □□□□□□□□□□□□□□□□□□□□□□□
vision_data = self.camera.detect_objects()
audio_data = self.microphone.recognize_speech()
return FusionResult(vision_data, audio_data, ...)
```

---

3. □□ AI □□AI Core□
□□□□□□□□□□□□□□□□□
```python
class HybridLogicNetwork:
def __init__(self):
self.symbolic_engine = SymbolicReasoner() # □□□□□□□□□□□/□□□□□
self.neural_model = TransformerModel() # □□□□□□□□□/□□□
self.emotion_model = EmotionPredictor() # □□□□□□

def reason(self, input_data):
# □□□□□□
symbolic_facts = self.extract_facts(input_data)
neural_output = self.neural_model.predict(input_data)
emotion_state = self.emotion_model.update(input_data)

# □□□□□□□□□□
if symbolic_facts.conflict_with(neural_output):
return self.resolve_conflict(symbolic_facts, neural_output)
else:
return self.generate_action(neural_output, emotion_state)
```

---

4. □□□□□□□
□□□□□□□□□□□
```python
class LanguageEmotionEngine:
def __init__(self):
self.nlp_pipeline = NLPModel("GPT-4") # □□□□□□□
self.emotion_graph = EmotionStateGraph() # □□□□□

def respond(self, input_text):
# □□□□□□□□□□
intent = self.nlp_pipeline.parse_intent(input_text)
emotion = self.emotion_graph.update(intent)

# □□□□□□□□□□□□□□+□□□□□
response_text = self.nlp_pipeline.generate(intent, emotion)
self.robot_face.set_expression(emotion)
self.speaker.play(response_text, tone=emotion.tone)
```

---

5. □□□□□□
□□□□□□□□□
```python
class DecisionMaker:
def __init__(self):
self.rl_agent = DQNAgent() # □□□□□□□□
self.task_planner = TaskPlanner() # □□□□□□□
```

```
def choose_action(self, state):
# [illegible]RL[illegible]
rl_actions = self.rl_agent.predict(state)
feasible_actions = self.task_planner.filter(rl_actions)
return self.utility_function.optimize(feasible_actions)
```

---

6. [illegible]

```python
class HumanoidRobot:
def __init__(self):
self.motors = MotorController()
self.sensors = SensorFusion()
self.ai = HybridLogicNetwork()
self.language = LanguageEmotionEngine()
self.decision = DecisionMaker()

def run_cycle(self):
while True:
sensor_data = self.sensors.update()
world_model = self.ai.reason(sensor_data)
action_plan = self.decision.choose_action(world_model)
self.motors.execute(action_plan)
```

[illegible]

1. **[illegible]**[illegible]ROS[illegible]MoveIt![illegible]
2. **[illegible]**[illegible]PyTorch/TensorFlow[illegible]OpenCV[illegible]
3. **[illegible]**[illegible]Prolog[illegible]Neural-Symbolic [illegible]
4. **[illegible]**[illegible]Whisper[illegible]VITS[illegible]
5. **[illegible]**[illegible]Arduino/ROS-Melodic[illegible]

[illegible]

- **[illegible]**[illegible] C++/Rust [illegible]
- **[illegible]**[illegible]
- **[illegible]**[illegible]

[illegible] AI [illegible]

●[illegible]**[illegible]**[illegible]**[illegible]**[illegible]**[illegible]

---

1. [illegible]

```python
```

```
# [illegible] ROS[illegible]
import rospy
from robotics_control import ArmController, HandGesture

class RoboticArm:
def __init__(self):
self.arm = ArmController()
self.hand = HandGesture()

def grasp_object(self, object_position):
# [illegible]
path = self.arm.calculate_trajectory(object_position)
self.arm.execute_motion(path)
# [illegible]
tactile_feedback = self.hand.read_sensors()
if tactile_feedback < threshold:
self.hand.adjust_grip(strength=0.8)
return "Grasp successful"

# [illegible]
robot_arm = RoboticArm()
robot_arm.grasp_object([x=0.5, y=0.2, z=1.0])
```

---

2. [illegible]/[illegible]/[illegible]

```python
# [illegible]
import tensorflow as tf
from sensors import Camera, Microphone, TactileSensor

class PerceptionModule:
def __init__(self):
self.vision_model = tf.keras.models.load_model('yolo_v7.h5')
self.audio_model = tf.keras.models.load_model('speech2text.h5')

def process_environment(self):
# [illegible]
image = Camera.capture()
objects = self.vision_model.predict(image)
# [illegible]
audio = Microphone.record()
speech_text = self.audio_model.predict(audio)
# [illegible]
touch_data = TactileSensor.read_pressure()
return {"objects": objects, "speech": speech_text, "touch": touch_data}
```

---

3. [illegible]

```python
# [illegible] + [illegible]
```

~~~
class ReasoningEngine:
def __init__(self):
self.knowledge_graph = load_knowledge_base("world_facts.ttl")
self.nn_model = torch.load("deep_reasoner.pth")

def hybrid_reasoning(self, input_query):
# [illegible] Prolog [illegible]
symbolic_result = self.symbolic_solver(input_query)
# [illegible]
nn_result = self.nn_model.predict(input_query)
# [illegible]
final_decision = self.fuse_results(symbolic_result, nn_result)
return final_decision

def fuse_results(self, logic_result, nn_result):
# [illegible]
if logic_result.confidence > 0.9:
return logic_result
else:
return nn_result
```

---

### **4. [illegible]**
```python
# [illegible] Transformer [illegible]
from transformers import GPT4, EmotionClassifier

class EmotionalAgent:
def __init__(self):
self.language_model = GPT4()
self.emotion_model = EmotionClassifier()

def respond(self, user_input):
# [illegible]
emotion = self.emotion_model.predict(user_input)
# [illegible]
response = self.language_model.generate(
prompt=user_input,
emotion=emotion, # [illegible]
max_length=100
)
# [illegible]TTS[illegible]
play_audio(tts_converter(response))
return response
```

---

### **5. [illegible]**
```python
# [illegible] PyTorch [illegible]
import torch
~~~

```
import torch.nn as nn

class CognitiveNN(nn.Module):
def __init__(self):
super().__init__()
# [illegible]
self.shared_encoder = TransformerEncoder()
# [illegible]
self.motion_head = MotionPredictor()
self.emotion_head = EmotionPredictor()
self.logic_head = LogicReasoner()

def forward(self, sensory_input):
features = self.shared_encoder(sensory_input)
motion = self.motion_head(features)
emotion = self.emotion_head(features)
logic = self.logic_head(features)
return {"motion": motion, "emotion": emotion, "logic": logic}
```

[illegible]
1. **[illegible]**[illegible]
2. **[illegible]**[illegible]
- [illegible] OpenAI [illegible] Dactyl [illegible]
- [illegible] Occupancy Networks[illegible]
3. **[illegible]**[illegible]
- [illegible] Prover9[illegible]
- [illegible]Markov [illegible]
4. **[illegible]**[illegible]
- [illegible] PAD [illegible]
- [illegible]

[illegible]
- **[illegible]**[illegible]ROS, MoveIt, Gazebo [illegible]
- **AI [illegible]**[illegible]TensorFlow, PyTorch, JAX
- **[illegible]**[illegible]Prolog, Datalog
- **[illegible]**[illegible]OpenCog, Hanson Robotics [illegible] Sophia [illegible]

[illegible]

Puppet show large-scale musical ball and high-intelligent robot large-scale musical ball ● Discuss the core technical features and functions of high-intelligent robots (or artificial intelligence systems). This kind of robot or system usually has a highly complex technical architecture, involving cutting-edge technologies in many fields. The following are the detailed explanations of these technical features:-1. The dexterous hand and limb movements. The movement ability of the intelligent robot is one of its important technical features, which is mainly reflected in the following aspects: (1) Multi-degree-of-freedom joint design: the robot's hands and limbs are usually equipped with joints with multiple degrees of freedom, which can simulate the human movement ability and realize complex movements, such as grasping, operating tools, walking and running. Driving technology: The motion of joints is usually driven by high-precision motors or steering gears, and these drivers can accurately control the angle and speed of joints. Force feedback and control: Through the sensor and force feedback system, the robot can sense the weight and surface characteristics of the object, thus realizing more natural and flexible grasping and operation. (2) Dynamic balance and coordinated motion planning: The robot can calculate and adjust the movements of limbs in real time through the motion planning algorithm to maintain balance and coordination. Real-time adjustment: In a complex environment, the robot can quickly adjust its posture, such as walking on uneven ground or moving quickly. -2. Developed brain neural network system The robot's "brain" is usually based on advanced neural network and artificial intelligence technology, which can realize a variety of complex functions: (1) Reasoning and thinking logic reasoning: robots can make reasoning and decisions through formal logic and mathematical logic. For example, it can solve complex mathematical problems or logical puzzles. Deep learning: Through deep learning algorithms, robots can learn patterns and laws from a large number of data, thus continuously optimizing their decision-making ability. (2) Language Ability Natural Language Processing (NLP): Robots can understand and generate natural languages and have a smooth dialogue with humans. Multilingual support: Some advanced robots can support multiple languages and even perform language translation. (3) Emotion and Expression Emotion Recognition: Robots can recognize human emotions through pronunciation and intonation, facial expressions and body language. Emotional expression: Robots can also express emotions, such as emotions, such as emotions, voices and body movements. (4) Complex Behavior Singing and Performance: The robot can realize complex behaviors such as singing and dancing through preset programs and sensor control. Play and Interaction: Some robots can simply play and interact with humans or other robots. -3. Logic system The logic system of the robot can be divided into multiple levels, from simple preliminary logic to complex deep logic reasoning: (1) Preliminary logic rule engine: the robot can make simple logic judgments and decisions based on preset rules. Conditional reflex: For example, when a specific signal is detected, the robot can perform a preset action. (2) Deep logic complex reasoning: The robot can make complex logic reasoning through deep learning and neural network. Mixed logic: Robots can combine various logic forms, such as formal logic, mathematical logic and mixed logic, to solve complex problems. (3) Multi-modal fusion of hybrid logic: The robot can fuse visual, auditory, tactile and other sensory information to form a more comprehensive logical judgment. Dynamic adjustment: In a complex environment, the robot can dynamically adjust the logical reasoning process according to real-time data. -4. Technical realization and application scenarios The realization of these technologies depends on cutting-edge technologies in many fields, including but not limited to: hardware: high-performance processors, sensors, motors and steering gears. Software: deep learning framework, natural language processing algorithm, motion planning algorithm. Application scenario: Robots can be used in education, entertainment, industry, medical care and other fields. -Summarize that highly intelligent robots have achieved highly flexible movement ability and powerful logical reasoning ability through complex hardware design and advanced software algorithms. They can not only simulate human body movements, but also interact naturally with human beings through emotional recognition and expression. The development of this technology is constantly promoting the application and popularization of robots in various fields. ● How do puppets dance and sing? Shake their heads and cry? Turn their eyes? How do puppets dance and sing? Shake their heads and cry? Turn their eyes? How do puppets' five fingers move and change? Like people, their neck, legs, joints and limbs move, their elbows and hands are equally important, their brains move, their hair flutters, their eyebrows change and their tongues spit out. The realization of puppet dancing, singing, shaking his head, wagging his tail, crying and rolling his eyes mainly depends on the puppet master's control skills and the design structure of the puppet itself. A detailed explanation of how puppets realize these movements: Mechanical structure of dance movements: Puppets usually consist of multiple movable joints, which are connected by wires, rods or other mechanical devices. By pulling or pushing these devices, the puppeteer makes the limbs and body of the puppet move according to the predetermined dance movements. Programming control: In some modern puppet performances, electronic equipment and programming may be used to control the movements of puppets. Puppets can perform complex dance action sequences through pre-set programs. Singing mouth movements: Puppets' mouths can usually be opened and closed. Puppet artists can synchronize the opening and closing of puppets' mouths with the rhythm of music through manual control or mechanical devices to simulate singing movements. Voice coordination: The actual voice is usually provided by voice actors or singers, and their voices are played through audio equipment, which is synchronized with the puppet's movements, creating the illusion that the puppet is singing. Shaking head joint: There is a movable joint between the puppet's head and its body. The puppet master can control the left and right rotation of the head by hand or mechanical device to realize the action of shaking his head. Balance of counterweight: In order to make the shaking of the head more natural, the head and neck of the puppet may be designed with appropriate counterweight to ensure the balance and stability of the puppet when shaking its head. Tail design: If a puppet has a tail, the tail is usually movable. The puppeteer swings the tail from side to side by pulling or pushing the wire or rod connecting the tail. Dynamic expression: In the performance, wagging the tail can be combined with the puppet's overall movements and expressions to enhance the puppet's dynamic expression and emotional communication. Crying facial expression: Puppet's face can be designed as changeable expression. By changing different facial parts or using movable facial features (such as eyes, eyebrows, mouth, etc.), the expression changes when crying can be simulated. Prop aid: In some cases, props may be used to enhance the crying effect, such as installing a "tear" device on the puppet's eyes, or using special lighting and sound effects to create a sad atmosphere. Mechanical device for rotating eyes: Puppet's eyes can be mounted on a rotatable device, and the puppeteer can rotate the eyes left and right through the control device to increase the agility and expressiveness of the puppet. Eye expression: Turning eyes can not only simulate human eye movement, but also convey the mood and attention of puppets through eye changes, making the performance more vivid and infectious. When performing puppet shows in residential areas and other places, the coordination and cooperation of these movements requires the puppeteer to have superb skills and rich performance experience, and at the same time, he should make flexible adjustments and innovations according to specific performance scenes and story lines to bring wonderful puppet performances to the audience. Puppet performance is a comprehensive art form, which requires a variety of equipment and props to complete a wonderful performance. The following are some basic puppet performance equipment: Puppet stick-head puppet: the action of the puppet is controlled by holding a pole. Marionette: Control the action of the puppet by pulling the thin thread connected to its joint. Bag puppet: the performer's gloves enter the puppet and directly control the puppet's movements with his hands and arms. Siamese Puppet: Performers wear special costumes and perform together with puppets. Control device lever: used to control the movements of the puppet's head, limbs and other parts. Rope lifting device: including rope, pulley, controller, etc., used to control the marionette. Remote control equipment: In some modern puppet shows, wireless remote control is used to control the puppet movements. Stage equipment Stage frame: A platform for performance can be designed into different shapes and sizes according to the needs of performance. Background scenery: including background curtain, scene props, etc., used to create the environment and atmosphere of the performance. Lighting equipment: used for lighting and creating different light and shadow effects to enhance the visual impact of the performance. Audio equipment: including speakers, microphones, etc., used to play music, sound effects and actors' dubbing. Other props, costumes and accessories: according to the role and performance content of the puppet, design and make suitable costumes and accessories for it. Props: such as furniture, tools, weapons, etc., are used to match the performance of puppets and increase the realism of the story. Cosmetic and painting supplies: used for facial makeup and body painting of puppets to make their images more vivid. These equipments are the basis of puppet performance, and different performance forms and styles may need some special equipment and props. The difficulty of remote puppet performance varies with many factors. Generally speaking, it is a challenging art form, but through proper training and skill mastery, these difficulties can be overcome and wonderful performances can be achieved. Detailed analysis of this problem: the technology requires high control accuracy: the remote control puppet needs to accurately control every movement of the puppet, from simple shaking head and wagging tail to complex dance movements, and the performer needs to have high control skills. This requires performers to have a deep understanding of the puppet structure and remote control equipment, and be able to skillfully operate various organs and devices. Coordination requirements: During the performance, the performer needs to control multiple parts of the puppet at the same time, such as the head, limbs and tail, which requires good hand-eye coordination and body coordination. For example, when manipulating a puppet to dance, it is a big challenge for beginners to keep the balance of the puppet's body and coordinate the movements of its limbs. Learning to master the basic movements with steep curves: Beginners need to spend a lot of time getting familiar with the basic movements of puppets, such as forward, backward, turning and waving. These seemingly simple movements actually need constant practice to be smooth and natural. Advanced complex movements: After mastering the basic movements, it is a long and difficult process to learn more complex movements and performance skills. For example, in order for a puppet to complete a coherent dance performance, it is necessary to skillfully combine several basic movements and adjust them according to the rhythm and emotion of music, which requires the performer to have certain artistic accomplishment and creativity. High artistic expression is required for the comprehensive quality of performers: remote puppet performance is not only a technical activity, but also an art. Performers need to convey emotions and stories through puppet movements, expressions and sounds. This requires performers to have good artistic perception and expressive force, and to be able to accurately express their inner feelings through puppets. Resilience: During the performance, you may encounter all kinds of unexpected situations, such as puppet dropping and equipment failure. Performers need to have a cool head and quick adaptability, and can take measures to solve problems in time to ensure the smooth performance. Equipment complexity increases the difficulty of equipment maintenance and debugging: the equipment of remote control puppet is more complicated, including remote control, receiver, battery and so on. Performers need to master the maintenance and debugging methods of the equipment to ensure the normal operation of the equipment during the performance. For example, the lack of battery power may affect the action effect of the puppet, which requires the performer to conduct a comprehensive inspection of the equipment before the performance. Signal interference: In some complex environments, there may be signal interference, which will affect the normal work of remote control equipment. Performers need to know how to avoid and solve the problem of signal interference to ensure the stability of puppet performance. The difficulty of integration with other art forms and the coordination of music: Puppet performances usually need to be closely combined with music to enhance the appeal of performances. Performers need to have a good sense of music and be able to adjust the speed and intensity of puppet movements according to the rhythm and melody of music, so that the two can be perfectly integrated. Cooperation with other performers: In some large puppet shows, there may be multiple performers manipulating different puppets at the same time. This requires good tacit understanding and teamwork spirit among performers, and through constant rehearsal and communication, the overall performance can be coordinated and unified. To sum up, the remote puppet show does have some difficulties, but through systematic training, continuous learning and rich practical experience, performers can gradually overcome these difficulties, improve their performance level and bring wonderful puppet shows to the audience. ● The main equipment needed for remote control puppet performance: the core control equipment remote control: it can usually be a smart phone or tablet computer with a special APP installed, and these devices send wireless signals to control the action of the puppet. Receiver: installed on the puppet or stage equipment, used to receive the signal sent by the remote controller and convert it into control instructions. Motor and steering gear: the key components to drive the puppet. The motor can control the overall movement of the puppet, such as forward, backward, turning, etc. The steering gear is used to accurately control the rotation of puppet joints, such as the movement of the head, limbs and other parts. Puppet body and accessories Puppet body: Puppet designed and manufactured according to performance requirements, and its internal structure needs to reserve space for installing equipment such as motors and steering gears. Joint connectors: such as cotton thread, connecting rod, etc., are used to transfer the power of steering gear to each joint of the puppet to realize flexible action control. Clothing and props: costumes, ornaments and props, such as weapons and tools, designed for puppets to enhance the expressive force of characters and the credibility of stories. Auxiliary equipment power supply equipment: including batteries, chargers, etc., which provide stable power support for the puppet's motor, steering gear and remote control equipment. Stage equipment: such as movable puppet hanging support frame and theater frame, which is used to build the stage environment for the performance and ensure that the puppet can perform stably. Multimedia equipment: including voice player, video recorder, LED strip, etc. It is used to play music, sound effects, dialogues, create stage lighting effects, and enhance the audio-visual experience of performances. Other optional equipment sensors, such as puppet sensor and digital compass, can enhance the interaction between the puppet and the audience, or be used for positioning and posture perception of the puppet. Information processing control center: such as Internet server or embedded system server, which is used to process complex interactive commands and data and realize multi-person interaction or remote control function. Through the cooperative work of these devices, the remote puppet show can realize a variety of actions and expressions, bringing wonderful visual enjoyment to the audience. Several common methods of connecting the motor and the steering gear with the puppet: connecting the head with the connecting rod: fixedly connecting the puppet head with the steering gear through the connecting rod to make the head a fixed stress point, and controlling the steering gear to realize the rotation of the head. For example, in some puppet performances, in this way, the puppet's head can be turned left and right or nodded up and down to enhance the puppet's expression and action expression. The limbs are connected with the mechanical manipulator: for the stick-head puppet, the mechanical manipulator can be used to connect the steering gear and the limbs of the puppet. One end of the mechanical control arm is connected.